EPFL

RLMEval: Evaluating Autoformalization on Research-Level Mathematics

Auguste Poiroux
Antoine Bosselut
Viktor Kunčak

AITP 2025

Recent results

IMO 2024:

- DeepMind[1]: 4/6 (formal)

IMO 2025:

- DeepMind[2] & OpenAI[3]: 5/6 (natural language)
- Harmonic[4]: 5/6 (formal)
- ByteDance[5]: 4+1/6 (formal)

[1]https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/
[2]https://deepmind.google/discover/blog/advanced-version-of-gemini-with-deep-think-officially-achieves-gold-medal-standard-at-the-international-mathematical-olympiad/
[3]https://github.com/aw31/openai-imo-2025-proofs/
[4]https://harmonic.fun/news
[5]*Seed-Prover: Deep and Broad Reasoning for Automated Theorem Proving*, ByteDance

High-School - MiniF2F

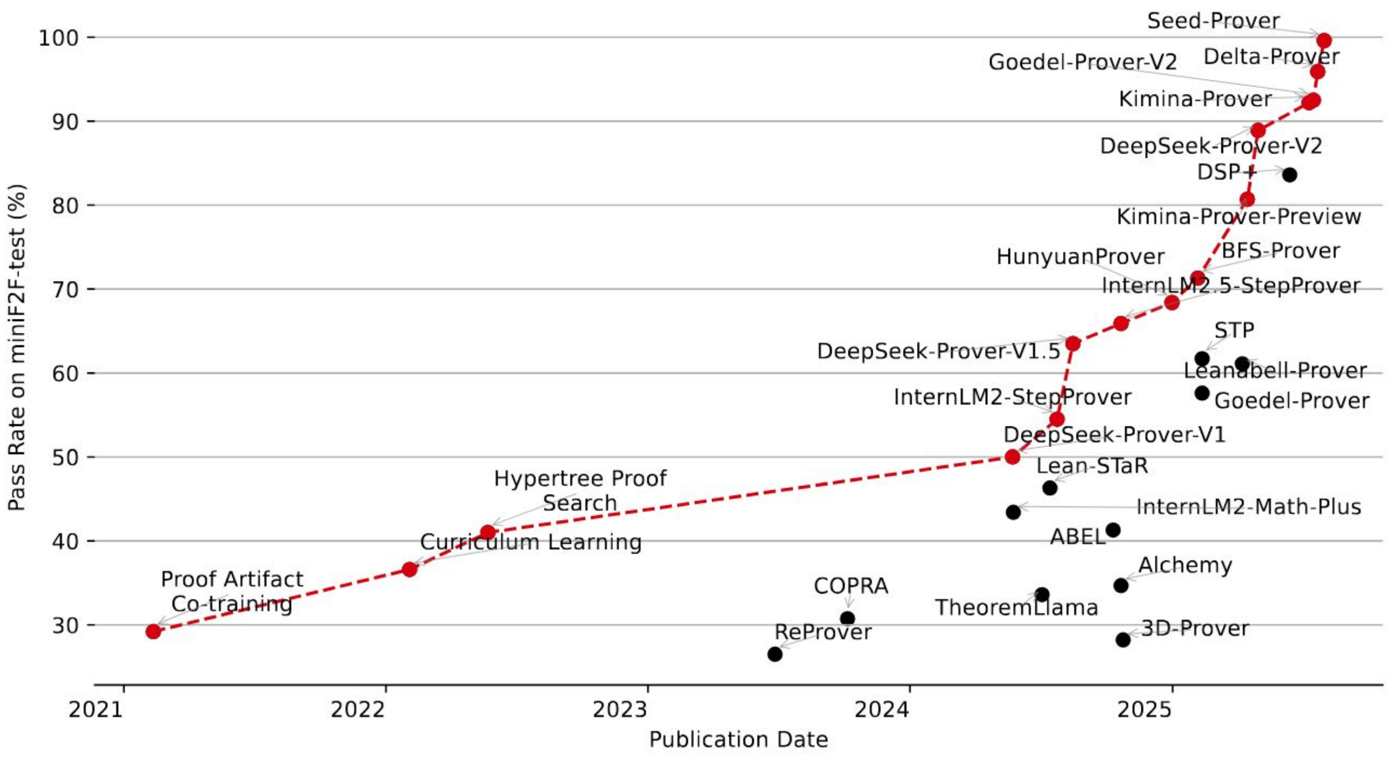
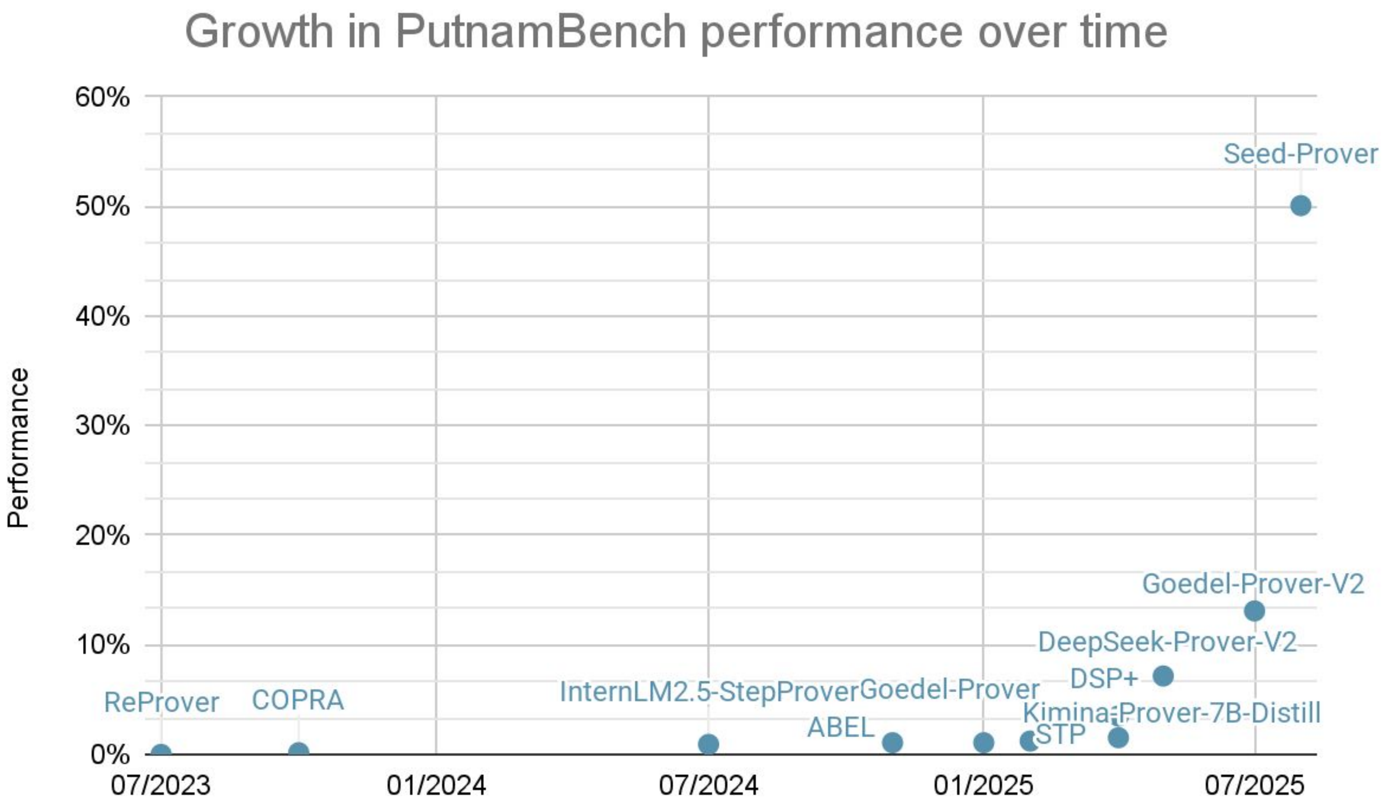


Figure 1 Growth in MiniF2F-Test performance over time.

Seed-Prover: Deep and Broad Reasoning for Automated Theorem Proving, ByteDance

Undergraduate - PutnamBench



Practical usage

SOTA models are not easily accessible

- Some of them are private: AlphaProof, SeedProver, …
- The others require, at least, high-end consumer GPUs
- SOTA results require tons of compute

Models struggle on real-world formalization problems

How do we measure and optimize for real-world use cases?

EPFL

Benchmark

Existing benchmarks:

- High-School and Undergraduate levels
- Standalone theorems & competition problems
  - i.e. no local dependencies
- Focused on theorem proving. What about autoformalization?

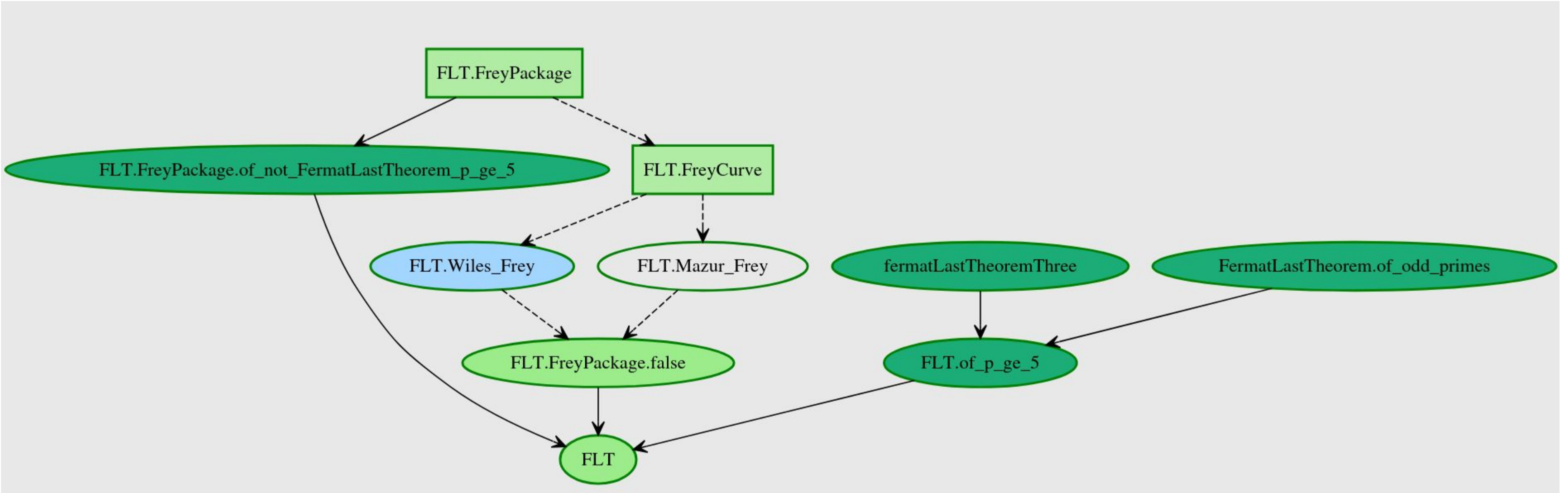
Challenges when curating benchmarks:

- Requires domain expertise
- Historically, formalization mistakes are frequent in AI benchmarks
  - ex: in ProofNet, >30% of the formal statements are invalid[1]

Can we leverage existing formalization projects?

[1] *Improving Autoformalization using Type-Checking*, Poiroux et al.

Lean Blueprint



Fine-grain alignment between natural language and Lean4 code

Used for formalizing research results:

- Polynomial Freiman-Ruzsa Conjecture, Tao et al. https://teorth.github.io/pfr/
- Carleson's theorem, van Doorn et al. https://florisvandoorn.com/carleson/
- Sphere Packing, Viazovska et al. https://thefundamentaltheor3m.github.io/Sphere-Packing-Lean/
- Medium Prime Number Theorem, Kontorovich et al. https://alexkontorovich.github.io/PrimeNumberTheoremAnd/web/

RLMEval

RLMEval: a framework to transform blueprint projects into benchmarks

RLM25: an instance of RLMEval on 6 formalization projects

- Total of 619 (natural-language, lean4) aligned theorems

Project	Domain	#Thms
Carleson	Analysis	110
FLT	Number Theory	52
PFR	Combinatorics	144
PNT	Analytic Number Theory	99
FLT3	Number Theory	84
TLB	Information & Probability Theory	124

Samples

PFR sample - Example of a relatively uninformative informal proof without broader context

Name: `cond-trial-ent`
File: `PFR/ForMathlib/Entropy/Basic.lean`

Theorem. *If 'X, Y' are conditionally independent over 'Z', then 'H[X, Y, Z] = H[X, Z] + H[Y, Z] - H[Z]'.*

Formal statement:

```
lemma ent_of_cond_indep (hX : Measurable X) (hY : Measurable Y) (hZ : Measurable Z)
    (h : CondIndepFun X Y Z μ) [IsZeroOrProbabilityMeasure μ]
    [FiniteRange X] [FiniteRange Y] [FiniteRange Z] :
    H[⟨X, ⟨Y, Z⟩⟩ ; μ] = H[⟨X, Z⟩; μ] + H[⟨Y, Z⟩; μ] - H[Z; μ]
```

Informal proof:
Immediate from `conditional-vanish` and `conditional-mutual-alt`.

Formal proof:

```
have hI : I[X : Y | Z ; μ] = 0 := (condMutualInfo_eq_zero hX hY).mpr h
rw [condMutualInfo_eq hX hY hZ] at hI
rw [entropy_assoc hX hY hZ, chain_rule _ (hX.prod_mk hY) hZ, chain_rule _ hX hZ, chain_rule _
   hY hZ]
linarith [hI]
```

Carleson sample - Typical difficult entry of RLMEval

Name: `tile-sum-operator`
File: `Carleson/FinitaryCarleson.lean`

Theorem. *We have for all $x \in G \setminus G'$*

$$\sum_{\mathfrak{p} \in \mathfrak{P}} T_{\mathfrak{p}} f(x) = \sum_{s=\sigma_1(x)}^{\sigma_2(x)} \int K_s(x,y) f(y) e(Q(x)(y) - Q(x)(x))\, d\mu(y). \quad (1)$$

Formal statement:

```
theorem tile_sum_operator {G' : Set X} {f : X → ℂ}
  {x : X} (hx : x ∈ G \ G') : ∑ (p : P X), carlesonOn p f x =
  ∑ s in Icc (σ₁ x) (σ₂ x), ∫ y, Ks s x y * f y * exp (I * (Q x y - Q x x))
```

Informal proof:
Fix $x \in G \setminus G'$. Sorting the tiles $\mathfrak{p}$ on the left-hand-side of (1) by the value $\mathrm{s}(\mathfrak{p}) \in [-S, S]$, it suffices to prove for every $-S \leq s \leq S$ that

$$\sum \mathfrak{p} \in \mathfrak{P} : \mathrm{s}(\mathfrak{p}) = sT\mathfrak{p}f(x) = 0 \quad (2)$$

if $s \notin [\sigma_1(x), \sigma 2(x)]$ and

$$\sum \mathfrak{p} \in \mathfrak{P} : \mathrm{s}(\mathfrak{p}) = sT\mathfrak{p}f(x) = \int Ks(x,y) f(y) e(Q(x)(y) - Q(x)(x)), d\mu(y). \quad (3)$$

if $s \in [\sigma_1(x), \sigma_2(x)]$. If $s \notin [\sigma_1(x), \sigma 2(x)]$, then by definition of $E(\mathfrak{p})$ we have $x \notin E(\mathfrak{p})$ for any $\mathfrak{p}$ with $\mathrm{s}(\mathfrak{p}) = s$ and thus $T\mathfrak{p}f(x) = 0$. This proves (2). Now assume $s \in [\sigma_1(x), \sigma 2(x)]$. By `coverdyadic`, `subsetmaxcube`, `eq-vol-sp-cube`, the fact that $c(I_0) = o$ and $G \subset B(o, \frac{1}{4}D^S)$, there is at least one $I \in \mathcal{D}$ with $s(I) = s$ and $x \in I$. By `dyadicproperty`, this I is unique. By `eq-dis-freq-cover`, there is precisely one $\mathfrak{p} \in \mathfrak{P}(I)$ such that $Q(x) \in \Omega(\mathfrak{p})$. Hence there is precisely one $\mathfrak{p} \in \mathfrak{P}$ with $\mathrm{s}(\mathfrak{p}) = s$ such that $x \in E(\mathfrak{p})$. For this $\mathfrak{p}$, the value $T\mathfrak{p}(x)$ by its definition in `definetp` equals the right-hand side of (3). This proves the lemma.

Formal proof:

```
rw [P_biUnion, Finset.sum_biUnion]; swap
· exact fun s _ s' _ hss' A hAs hAs' p pA ↦ False.elim <| hss' (s_eq (hAs pA) ▹ s_eq (hAs' pA
    ))
rw [← (Icc (-S : ℤ) S).toFinset.sum_filter_add_sum_filter_not (fun s ↦ s ∈ Icc (σ₁ x) (σ₂ x
    ))]
rw [Finset.sum_eq_zero sum_eq_zero_of_nmem_Icc, add_zero]
refine Finset.sum_congr (Finset.ext fun s ↦ ⟨fun hs ↦ ?_, fun hs ↦ ?_⟩) (fun s hs ↦ ?_)
· rw [Finset.mem_filter, ← mem_toFinset] at hs
exact hs.2
· rw [mem_toFinset] at hs
rw [toFinset_Icc, Finset.mem_filter]
exact ⟨Finset.mem_Icc.2 (Icc_σ_subset_Icc_S hs), hs⟩
· rcases exists_Grid hx.1 hs with ⟨I, Is, xI⟩
obtain ⟨p, IpI, Qp⟩ : ∃ (p : P X), I p = I ∧ Q x ∈ Ω p := by simpa using biUnion_Ω ⟨x, rfl⟩
have pPXs : p ∈ PX_s s := by simpa [s, IpI]
have : ∀ p' ∈ PX_s s, p' ≠ p → carlesonOn p' f x = 0 := by
  intro p' p'PXs p'p
  apply indicator_of_not_mem
  simp only [E, mem_setOf_eq, not_and]
  refine fun x_in_Pp' Qp' ↦ False.elim ?_
  have s_eq := s_eq pPXs ▹ s_eq p'PXs
  have : ¬ Disjoint (I p' : Set X) (I p : Set X) := not_disjoint_iff.2 ⟨x, x_in_Ip', IpI ▹ xI⟩
  exact disjoint_left.1 (disjoint_Ω p'p <| Or.resolve_right (eq_or_disjoint s_eq) this) Qp'
    Qp
rw [Finset.sum_eq_single_of_mem p pPXs this]
have xEp : x ∈ E p :=
  ⟨IpI ▹ xI, Qp, by simpa only [toFinset_Icc, Finset.mem_Icc, s_eq pPXs] using hs⟩
simp_rw [carlesonOn_def', indicator_of_mem xEp, s_eq pPXs]
```

RLMEval - Evaluation tasks

Automated/Neural Theorem Proving

- Input: theorem to prove (+ local context)
- Output: formal proof

Proof Autoformalization

- Input: theorem to prove + informal proof (+ local context)
- Output: formal proof

Statement Autoformalization

- Input: informal theorem to formalize (+ local context)
- Output: formal theorem

EPFL

RLMEval - Evaluation tasks

Project	% Auxiliary lemmas	Main theorems Proof Length	Auxiliary lemmas Proof Length
PFR	75.3 %	23.2	9.0
FLT	72.2 %	12.8	4.0
FLT3	15.9 %	8.8	4.9
Carleson	85.9 %	27.0	7.8
PNT	78.3 %	16.7	8.3
TLB	83.0 %	11.2	5.7
Avg	68.3 %	16.6	6.6

Auxiliary lemma = Lean lemma not appearing in the *informal* blueprint

ATP/NTP + Proof autoformalization tasks:

- **Easy mode:** auxiliary lemmas are available in the context
- **Normal mode:** auxiliary lemmas are hidden from the evaluated model

RLM25 - Results

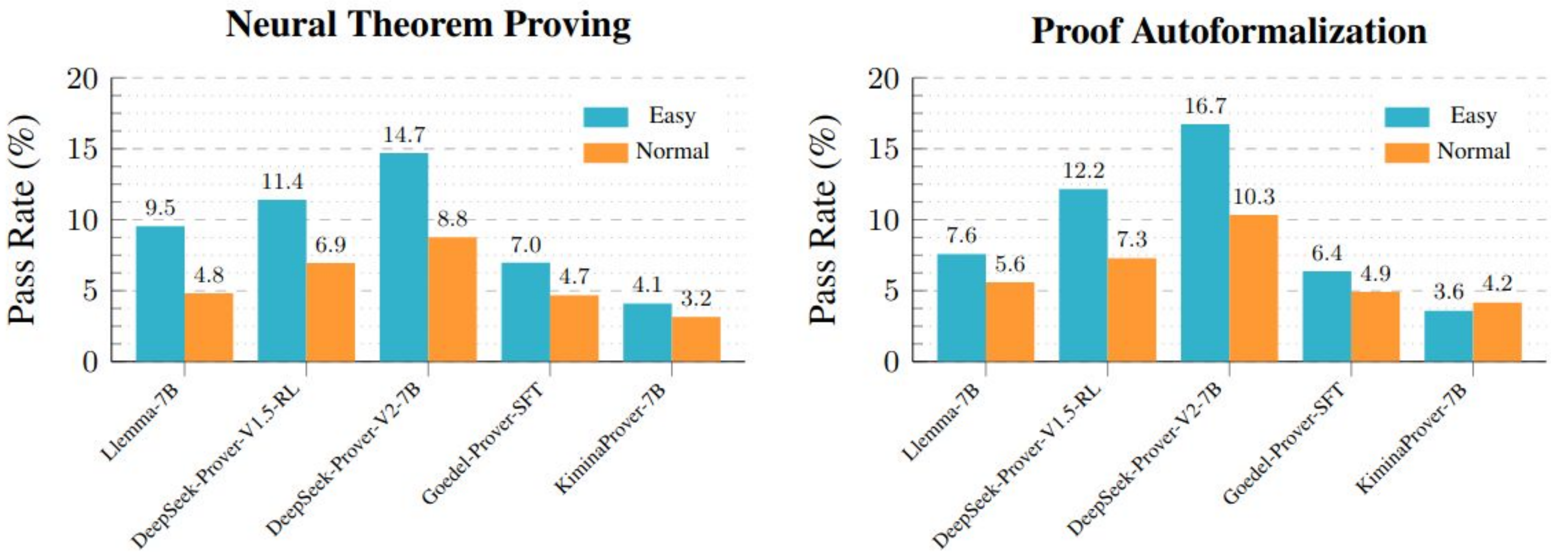
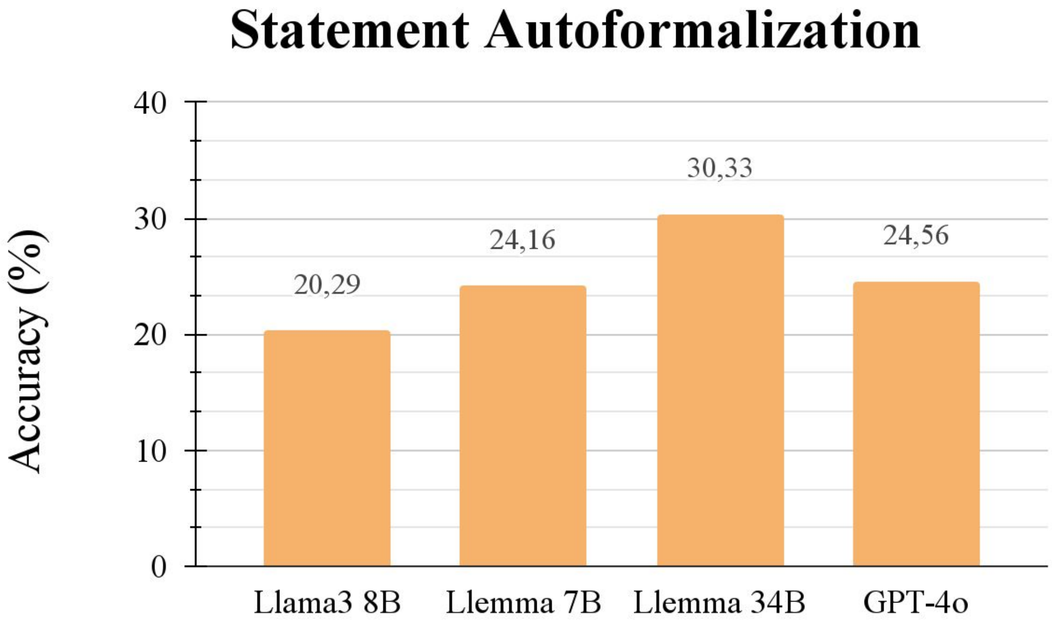


Figure 1: Pass rate on RLMEval using pass@128 for neural theorem proving (left) and proof autoformalization (right), in Easy and Normal modes.

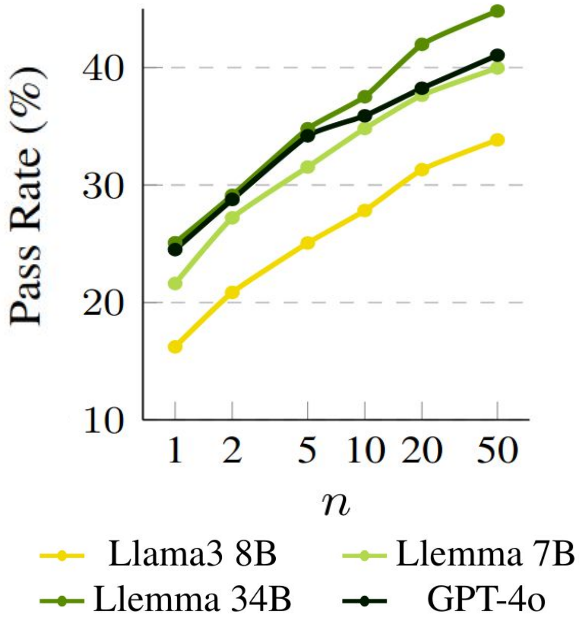
RLM25 - Results



Accuracy on RLM25 using in-file content prompting and n=50 samples + Self-BLEU for each informal theorem to formalize

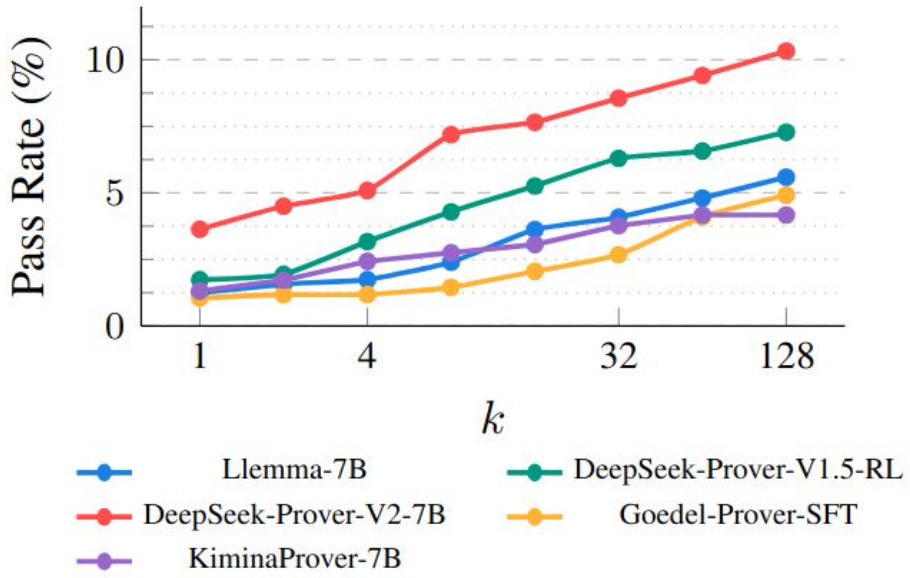
Scaling

Statement Autoformalization

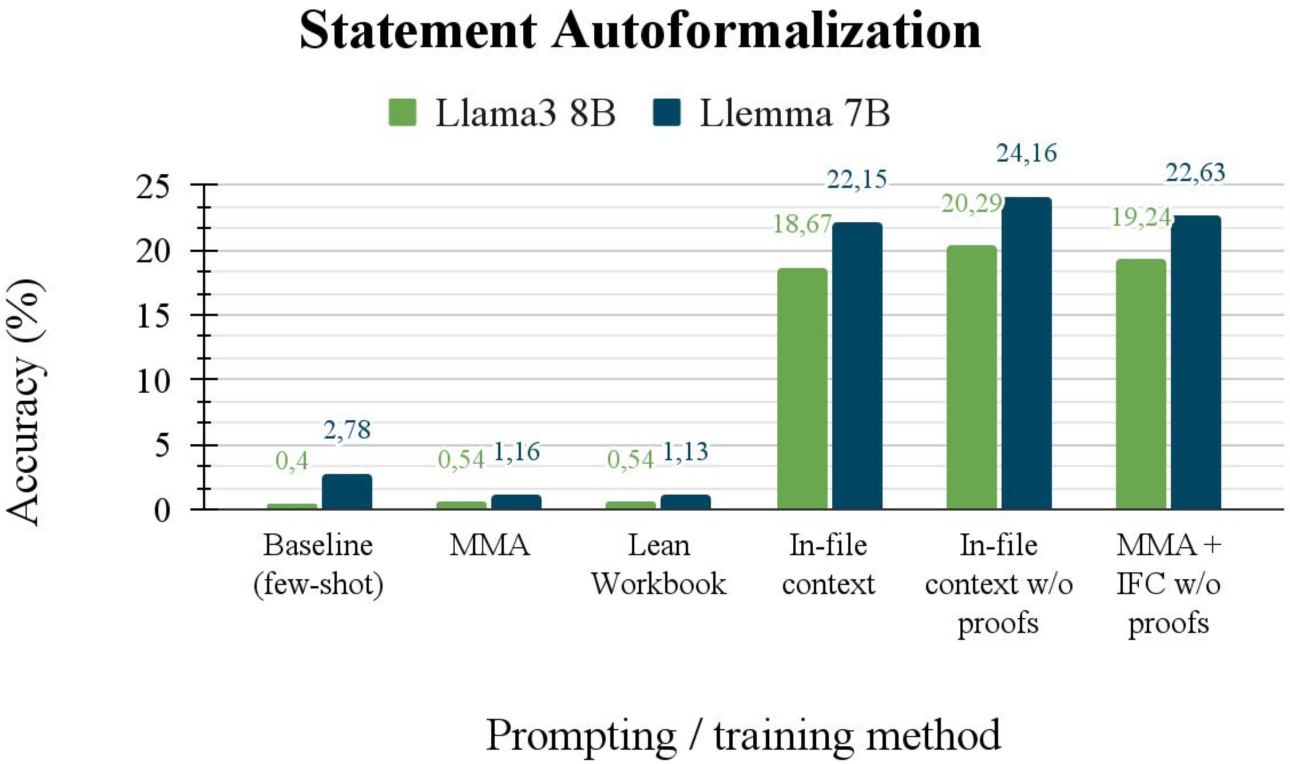


Proof Autoformalization

Normal mode



RLM25 - Case Study



EPFL

Next steps

RLMEval only captures a small part of real-world formalization tasks

Non-exhaustive list of capabilities to evaluate in the future:

- Deriving the informal blueprint from a paper
- Refining and adapting the blueprint/formalization during the formalization process
- Refactoring Lean code to make components reusable / integrated to Mathlib
- Definition formalization
- …

LeanInteract

Lean4↔Python interface

- Support Lean v4.7.0-rc1 to v4.23.0-rc2 (>50 versions)
- Spin up ephemeral Lean projects
- >10k downloads on PyPI

Backend:

- Lean REPL[1] fork
- Latest features & bug fixes semi-automatically backported

Coming soon: improved data extraction, incremental (Lean >= v4.8.0) & parallel elaboration (Lean >= v4.19.0)

[1] *Lean REPL,* Morrison et al., github.com/leanprover-community/repl

Conclusion

RLMEval:

- Measure performance on real-world projects
- Current training methods have limited success

→ project-wide context awareness is necessary

Papers:

- *Reliable Evaluation and Benchmarks for Statement Autoformalization*, Poiroux et al., **EMNLP 2025**
- *RLMEval: Evaluating Research-Level Neural Theorem Proving*, Poiroux et al., **EMNLP 2025 - Findings**

RLMEval

LeanInteract

Table 4: Detailed pass@k rates (%) for Proof Autoformalization on RLMEval projects. Normal mode uses on blueprint lemmas, Easy mode uses all project lemmas. Projects are: PFR, FLT3 (Fermat's Last Theorem for n=3 Carleson (Carl.), FLT (Fermat's Last Theorem), TLB (testing-lower-bounds), PNT (Prime Number Theorem And

Model	Mode	p@k	PFR	FLT3	Carl.	FLT	TLB	PNT	Total
Llemma 7B	Normal	p@1	0.69	3.57	0.00	0.00	3.23	0.00	1.25
		p@32	0.69	9.52	0.91	3.85	6.45	3.03	4.08
		p@128	0.69	14.29	0.91	5.77	8.87	3.03	5.59
	Easy	p@1	0.00	2.38	0.00	1.92	4.03	1.01	1.56
		p@32	0.69	9.52	0.91	9.62	6.45	3.03	5.04
		p@128	1.39	13.10	0.91	15.38	9.68	5.05	7.58
DeepSeek-Prover-V1.5-RL	Normal	p@1	0.00	2.38	0.91	1.92	3.23	2.02	1.74
		p@32	1.39	19.05	0.91	9.62	4.84	2.02	6.30
		p@128	2.08	22.62	0.91	9.62	6.45	2.02	7.28
	Easy	p@1	0.69	5.95	0.00	0.00	1.61	2.02	1.71
		p@32	0.69	16.67	0.91	19.23	12.90	8.08	9.75
		p@128	3.47	23.81	0.91	21.15	14.52	9.09	12.16
DeepSeek-Prover-V2-7B	Normal	p@1	0.69	11.90	0.00	5.77	2.42	1.01	3.63
		p@32	2.08	25.00	1.82	11.54	8.87	2.02	8.56
		p@128	2.08	32.14	2.73	11.54	10.48	3.03	10.33
	Easy	p@1	0.69	7.14	0.91	9.62	4.84	3.03	4.37
		p@32	3.47	27.38	1.82	23.08	19.35	10.10	14.20
		p@128	4.86	32.14	3.64	23.08	22.58	14.14	16.74
Goedel-Prover-SFT	Normal	p@1	0.00	3.57	0.00	1.92	0.81	0.00	1.05
		p@32	0.69	8.33	0.00	3.85	3.23	0.00	2.68
		p@128	0.69	13.10	0.00	9.62	4.03	2.02	4.91
	Easy	p@1	0.00	1.19	0.00	0.00	0.81	0.00	0.33
		p@32	0.00	8.33	0.00	13.46	4.84	2.02	4.78
		p@128	0.69	10.71	0.00	17.31	6.45	3.03	6.37
KiminaProver-7B	Normal	p@1	0.00	3.57	0.00	1.92	2.42	0.00	1.32
		p@32	0.00	10.71	0.00	7.69	3.23	1.01	3.77
		p@128	0.00	13.10	0.00	7.69	3.23	1.01	4.17
	Easy	p@1	0.00	3.57	0.00	1.92	0.00	0.00	0.92
		p@32	0.00	7.14	0.00	5.77	2.42	3.03	3.06
		p@128	0.00	9.52	0.00	5.77	3.23	3.03	3.59